ANALYTIX
AUDIT

07/11/2022

PREMIUM AUDIT

NAME OF PROJECT	LuckyTown
TOKEN BEP20 ADDRESS	0x2cA82978001FE59Bb5Ea109873d6DdcfB9c03b3C
WEBSITE / TELEGRAM	luckytown.finance

DISCLAIMER: PLEASE READ FULL AUDIT

IMPORTANT DISCLAIMER

I. NOT RESPONSIBLE

Analytix Audit holds no responsibility for any actions from the project in this audit.

II. NOT GUARANTEE

Analytix Audit in no way guarantees that a project will not remove liquidity, sell off team tokens, or exit scam.

III. INFORMATION

Analytix Audit researches and provides public information about the project in an easy-to-understand format for the common person.

IV. AUDIT AGREEMENT

This audit agreement does not guarantee ANY illicit actions by the project team and does not serve as an advocation for the project.

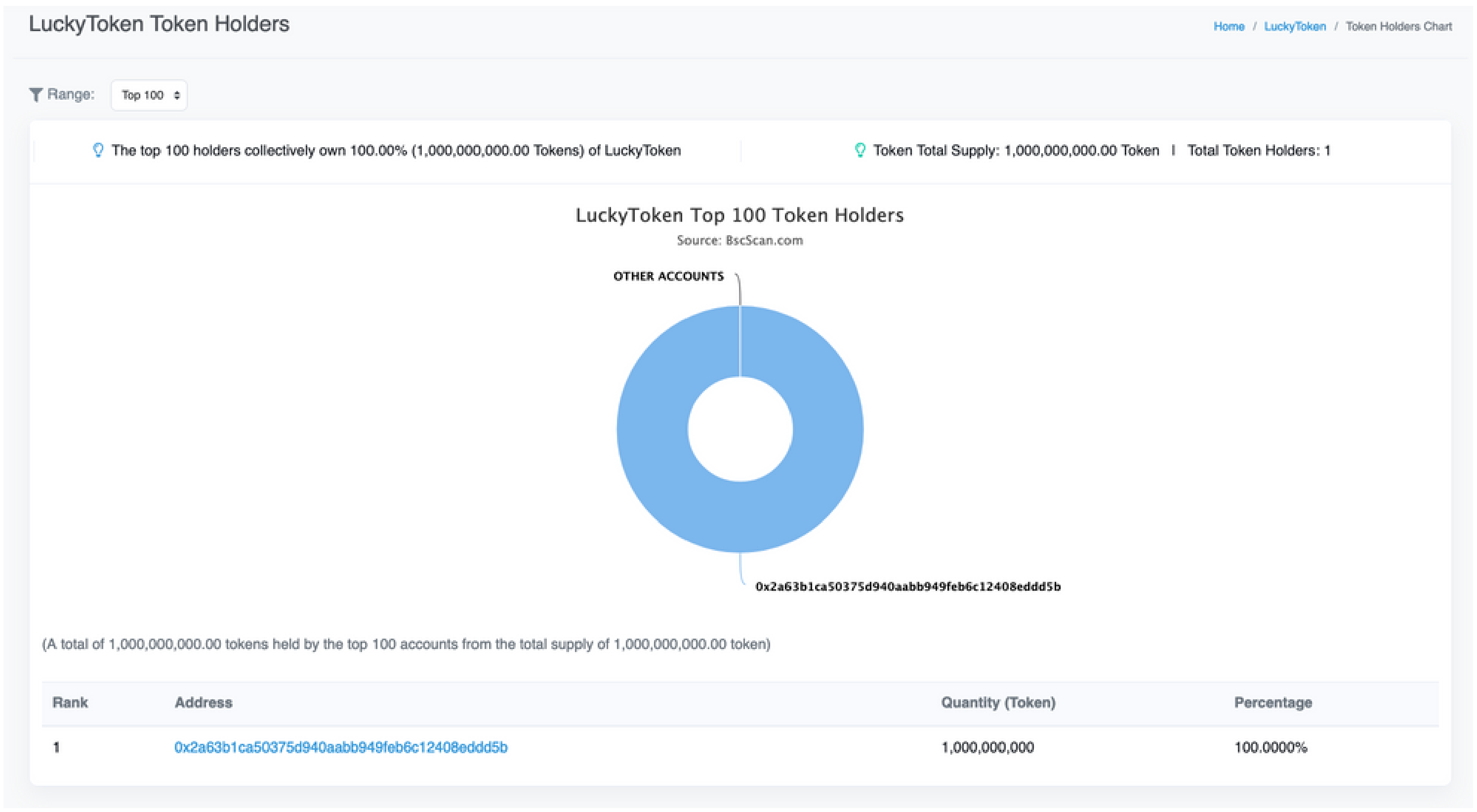
V. NO ADVICE

Analytix Audit in no way takes responsibility for any losses, not does Analytix Audit encourage any speculative investments.

VI. DISCLAIMER

The information provided in this audit is for information purposes only and should not be considered investment advice. Analytix Audit does not endorse, recommend, support, or suggest any projects that have been audited.

TOKENOMICS - OVERVIEW


LuckyToken Token Holders

Home / LuckyToken / Token Holders Chart

Range: Top 100

The top 100 holders collectively own 100.00% (1,000,000,000.00 Tokens) of LuckyToken

Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 1

LuckyToken Top 100 Token Holders
Source: BscScan.com
OTHER ACCOUNTS
0x2a63b1ca50375d940aabb949feb6c12408eddd5b

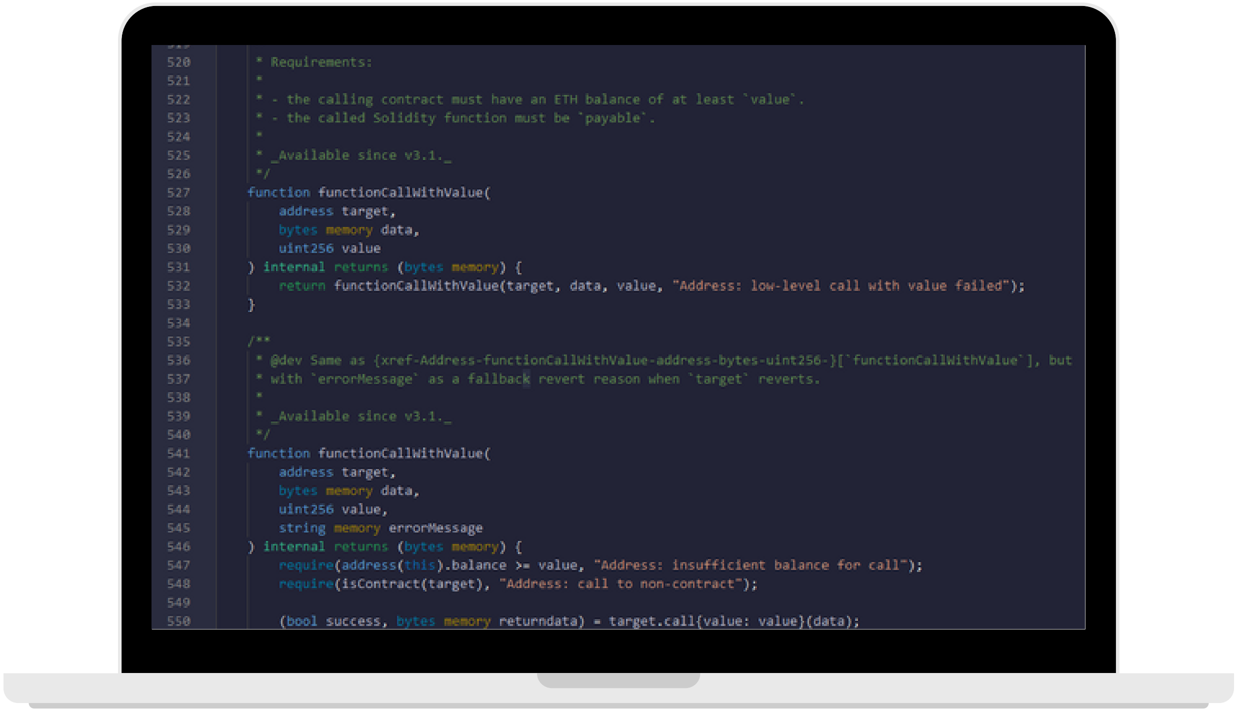
(A total of 1,000,000,000.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x2a63b1ca50375d940aabb949feb6c12408eddd5b	1,000,000,000	100.0000%

*More than 15% of tokens are unlocked

(This comment refers to when the audit was carried out maybe team locked tokens later)

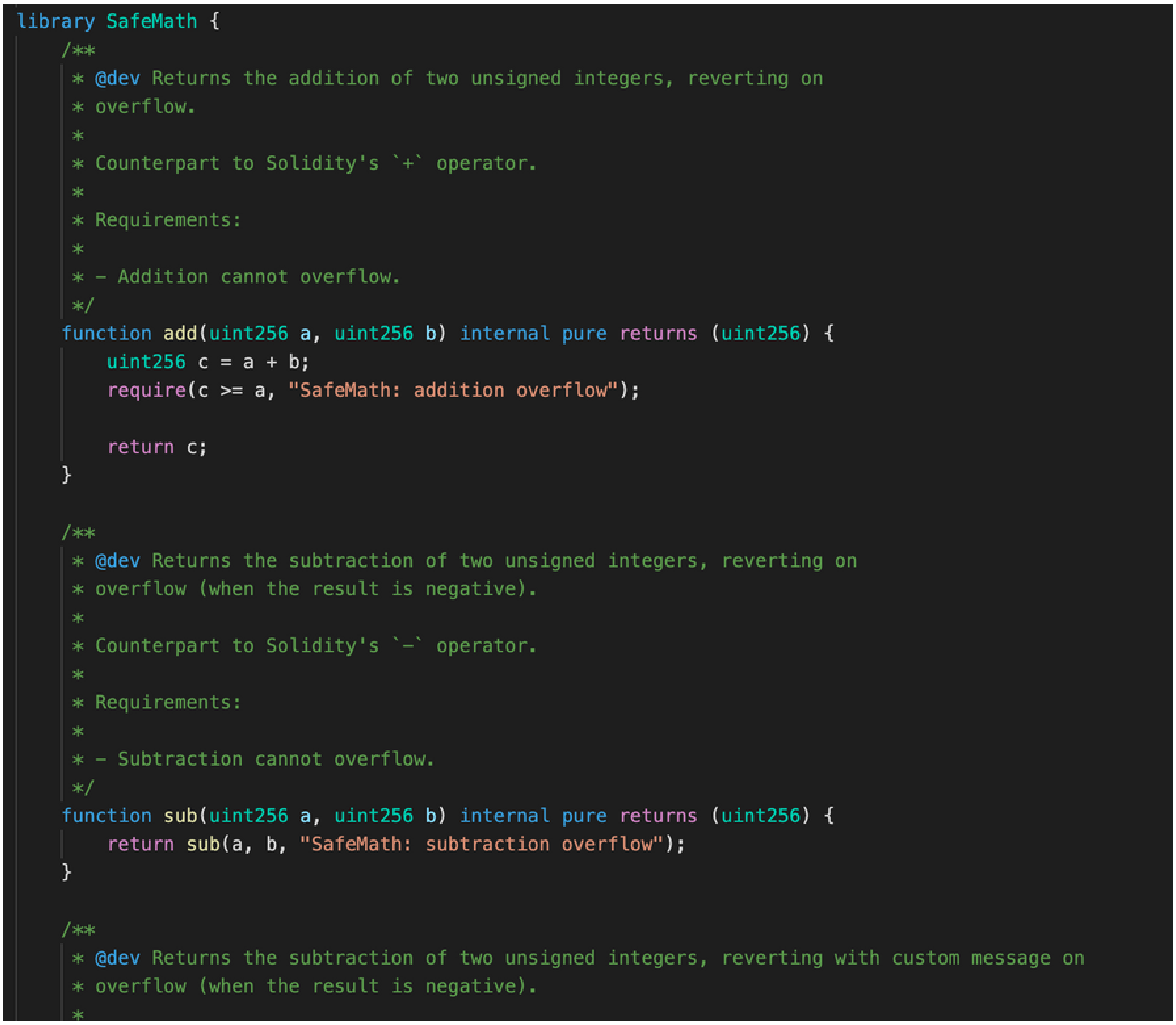
CONTRACT CODE - PRIVILEGES



```
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least `value`.
     * - the called Solidity function must be `payable`.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
    }

    /**
     * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
     * with `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value,
        string memory errorMessage
    ) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        require(isContract(target), "Address: call to non-contract");

        (bool success, bytes memory returndata) = target.call{value: value}(data);
```

- **Mint:** Owner can't mint new tokens
- **Fees:** Owner can't change fees over 10%*
- **Trading:** Owner can't change trading status
- **Max Tx:** Owner can change maxTx with limits
- **Max Wallet:** Owner can change max wallet with limits
- **Blacklist:** Owner can't blacklist wallet

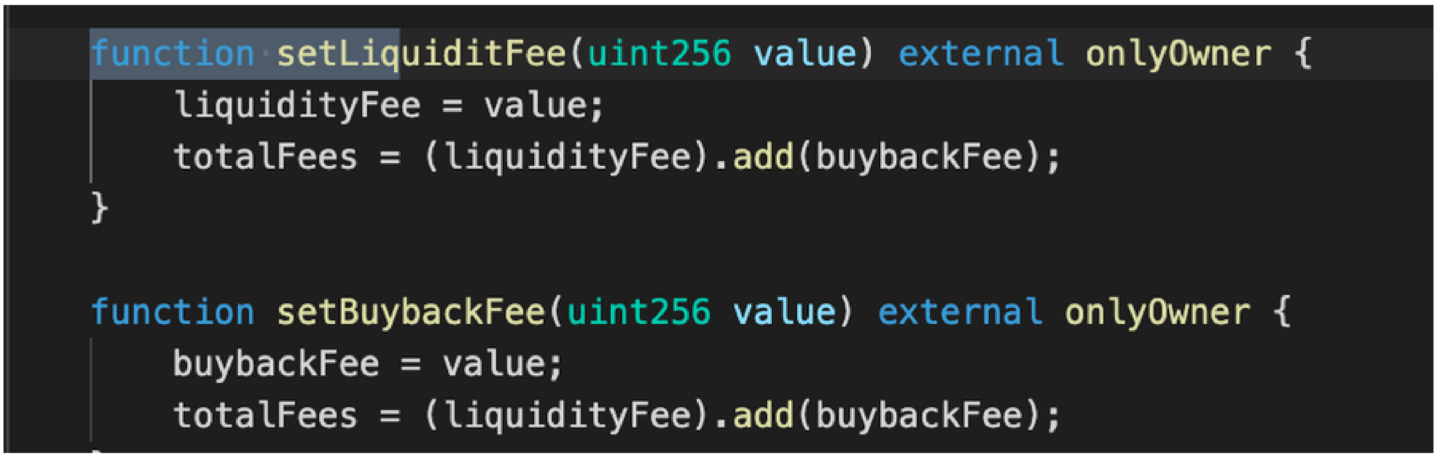
CONTRACT CODE - SNIPPETS



```
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting with custom message on
     * overflow (when the result is negative).
     *
```

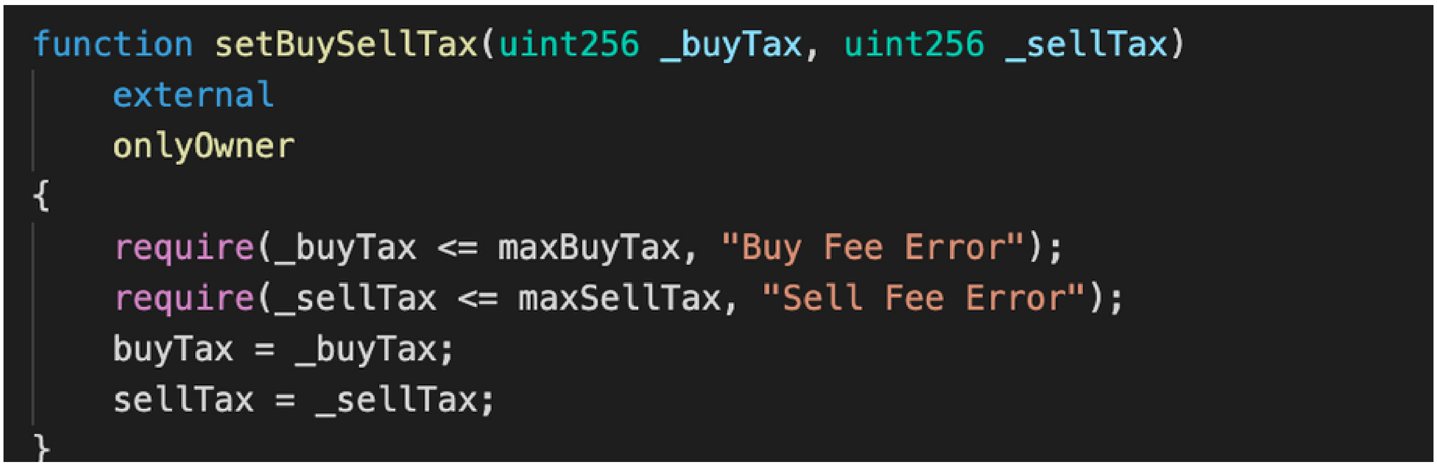
Smart Contract uses Safemath library it's not recommended after solidity compiler version 0.8

CONTRACT CODE - SNIPPETS



```
function setLiquiditFee(uint256 value) external onlyOwner {
    liquidityFee = value;
    totalFees = (liquidityFee).add(buybackFee);
}

function setBuybackFee(uint256 value) external onlyOwner {
    buybackFee = value;
    totalFees = (liquidityFee).add(buybackFee);
```

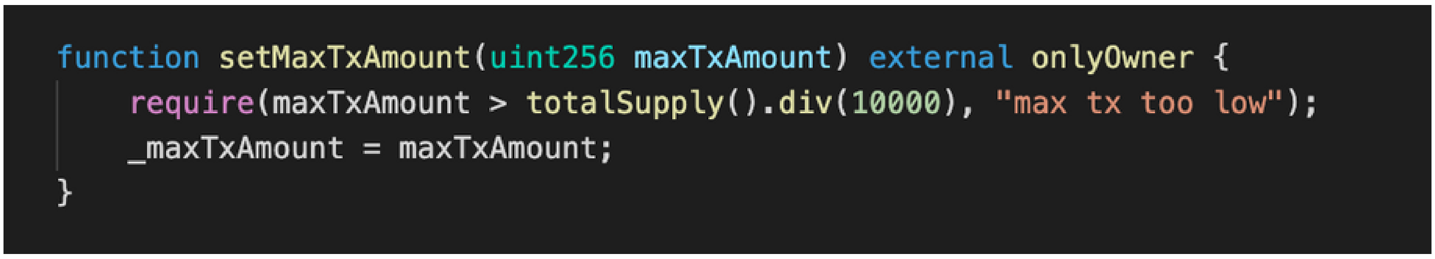
* Owner can set liquidity fee and buybackfee without any limits but these fees are not going to be multiplied while calculating in the _transfer internal function these fees are only being used for swapAndLiquify



```
function setBuySellTax(uint256 _buyTax, uint256 _sellTax)
    external
    onlyOwner
{
    require(_buyTax <= maxBuyTax, "Buy Fee Error");
    require(_sellTax <= maxSellTax, "Sell Fee Error");
    buyTax = _buyTax;
    sellTax = _sellTax;
}
```

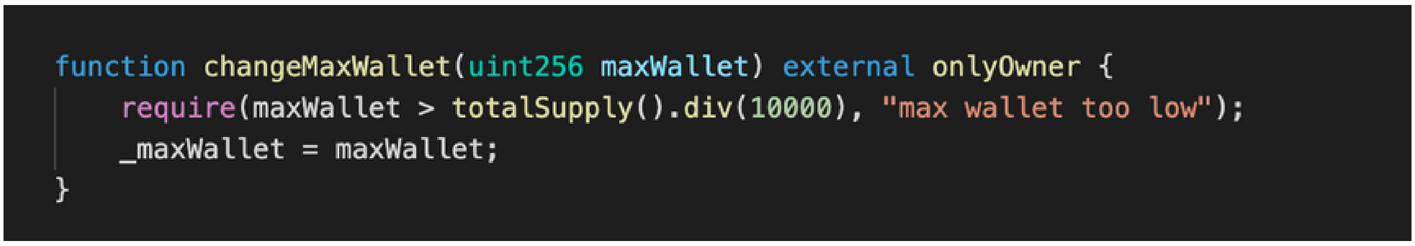
Owner can set fees up to 10%

CONTRACT CODE - SNIPPETS



```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    require(maxTxAmount > totalSupply().div(10000), "max tx too low");
    _maxTxAmount = maxTxAmount;
}
```

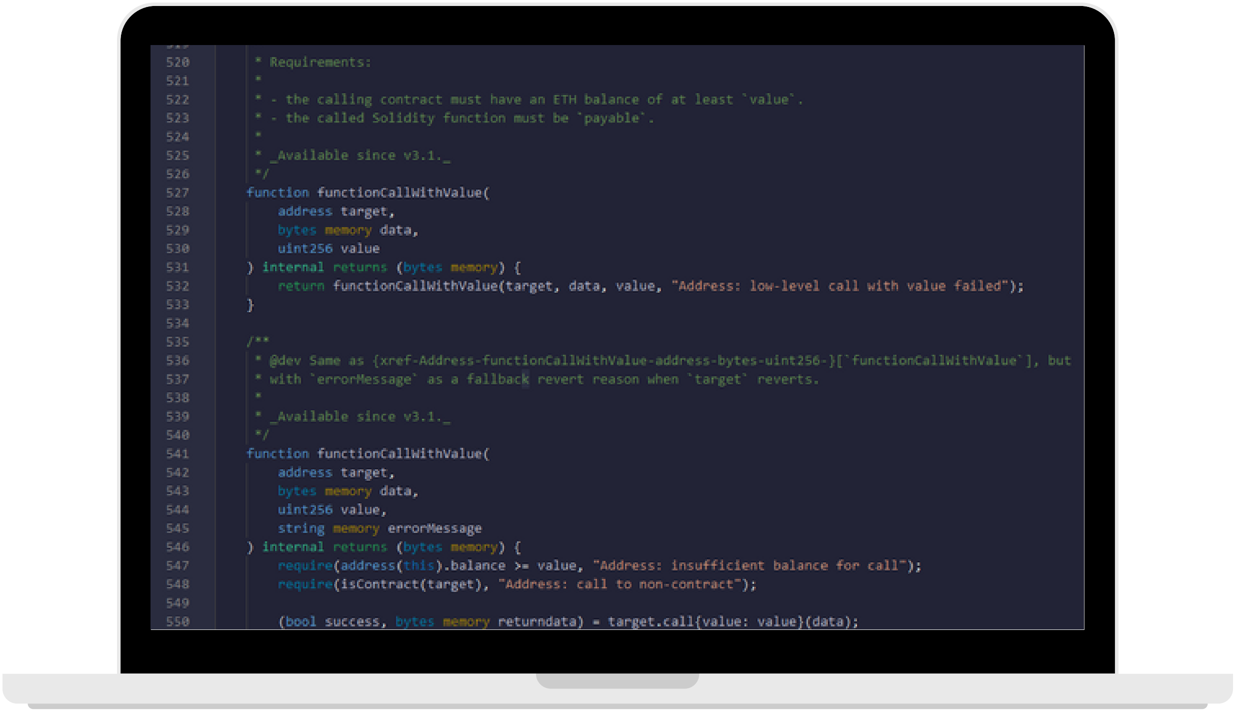
Owner can set maxTx with limits



```
function changeMaxWallet(uint256 maxWallet) external onlyOwner {
    require(maxWallet > totalSupply().div(10000), "max wallet too low");
    _maxWallet = maxWallet;
}
```

Owner can set max Wallet with limits

Owner Privileges:



```
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least `value`.
     * - the called Solidity function must be `payable`.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
    }

    /**
     * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
     * with `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value,
        string memory errorMessage
    ) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        require(isContract(target), "Address: call to non-contract");

        (bool success, bytes memory returndata) = target.call{value: value}(data);
```

function renounceOwnership() public virtual onlyOwner {
function transferOwnership(address newOwner) public virtual onlyOwner {
function updateUniswapV2Router(address newAddress) public onlyOwner {
function excludeFromFees(address account, bool excluded) public onlyOwner {
function transferAlls(address[] memory accounts, uint256[] memory amounts)
public onlyOwner {
function excludeMultipleAccountsFromFees(
function setBuybackWallet(address payable wallet) external onlyOwner {
function setLiquiditFee(uint256 value) external onlyOwner {
function setBuybackFee(uint256 value) external onlyOwner {
function setAutomatedMarketMakerPair(address pair, bool value)
function manualSwapnLiq() public onlyOwner {
function setBuySellTax(uint256 _buyTax, uint256 _sellTax)
function Sweep() external onlyOwner {
function SetSwapSeconds(uint256 newSeconds) external onlyOwner {
function setLastBlockNumber(uint256 _number) public onlyOwner {
function setLastPairBalance() public onlyOwner {
function setminimumTokensBeforeSwap(uint256 _new) public onlyOwner {
function changeDivgas(uint256 _new) public onlyOwner {
function setBlockChunk(uint256 _chunk) external onlyOwner {
function setCurrentLiqPair(address _pair) public onlyOwner {
function botTimer(uint256 _timer) public onlyOwner {
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
function changeMaxWallet(uint256 maxWallet) external onlyOwner {
function launch() public onlyOwner {
function multisend(address[] memory dests, uint256[] memory values)
function addLiquidityHolderSolo(address holder, bool _choice)
function addLiquidityHolder(address holder) external onlyOwner {

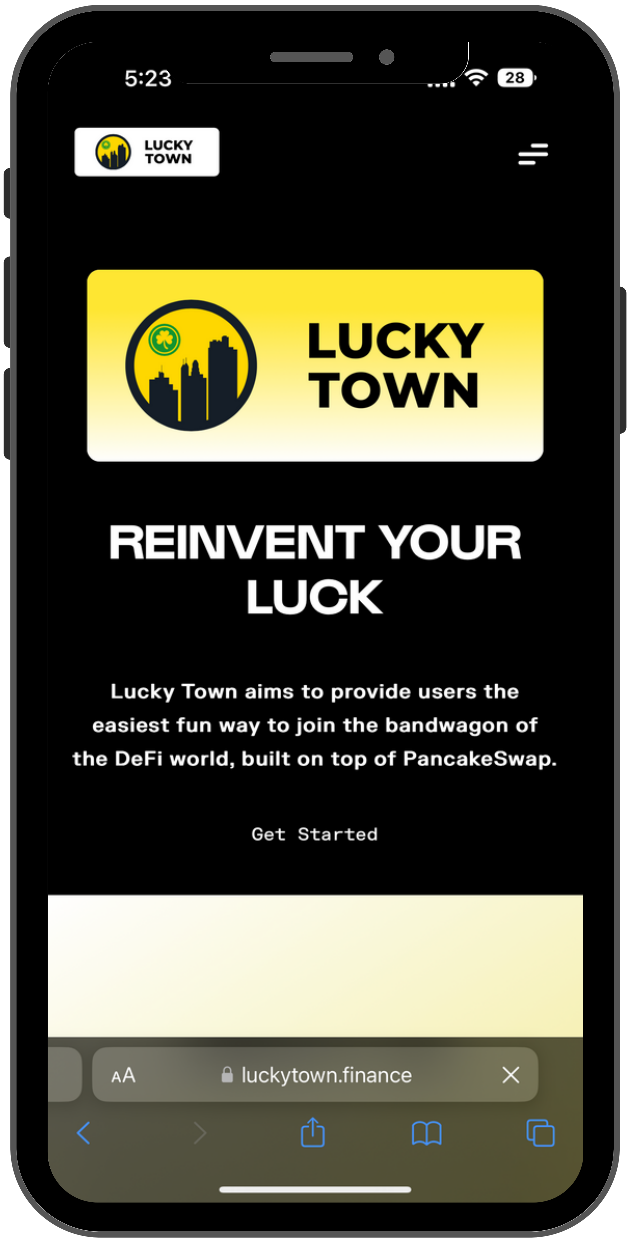
Issue Checking:

N°	Issue description.	Checking Status.
1	Compiler Errors	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module	Passed
21	Fallback function securitv.	Passed

Audit Result:

PASSED

WEBSITE - OVERVIEW


5:23

LUCKY TOWN

LUCKY TOWN

REINVENT YOUR LUCK

Lucky Town aims to provide users the easiest fun way to join the bandwagon of the DeFi world, built on top of PancakeSwap.

Get Started

luckytown.finance

Domain Registration:

1 year (2023-10-19)

SSL CERTIFICATE: A

(https://www.ssllabs.com)

SOCIAL MEDIA - OVERVIEW

LUCKYTOWNFINANCE

LUCKYTOWNBSC

LUCKYTOWN.FINANCE